

# Data Mining Exam

May 28, 2021

Kasper Rosenkrands

Aalborg University  
Denmark



**AALBORG UNIVERSITY**  
DENMARK

# Clustering

## Overview



- ▶ What is clustering?
- ▶ K-Means optimization problem and algorithm
- ▶ Implementation of the K-Means algorithm and an example
- ▶ Hierarchical Clustering (briefly)

# Clustering

## Introduction



**Clustering** is a way to categorize data to impose structure.

A use case is recommender systems (Amazon, Spotify, Netflix), where a user is recommended items that bought/listened to/watched by other users with similar interests.

# Clustering

## K-Means Optimization Problem

Given  $D = (x_1, \dots, x_n)$  where  $x_i \in \mathbb{R}^p$ ,  $K \in \mathbb{N}$  and let  $C_1, \dots, C_K$  denote different groups of the  $x_i$ 's.

The K-Means algorithm tries to solve

$$\min_{C_1, \dots, C_K} \left\{ \sum_{k=1}^K W(C_k) \right\}, \quad (1)$$

where  $W(C_k)$  denotes the **within cluster variation**, in other words the dissimilarity of the group.

The most common dissimilarity measure is the squared Euclidean distance

$$W(C_k) := \frac{1}{|C_k|} \sum_{i, i' \in C_k} \sum_{j=1}^p (x_{i,j} - x_{i',j})^2. \quad (2)$$

# Clustering

## K-Means Optimization Problem

If we by  $\bar{x}_{k,j} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{i,j}$  denote the mean value of the  $j$ 'th dimension in cluster  $k$ , it can be shown that

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^p (x_{i,j} - x_{i',j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^p (x_{i,j} - \bar{x}_{k,j})^2. \quad (3)$$

If we further note that  $\bar{x}_{k,j} = \min_{\mu_k} \left\{ \sum_{i \in C_k} \sum_{j=1}^p (x_{i,j} - \mu_k)^2 \right\}$  this implies that the optimization problem in (1) can be rewritten as

$$\min_{C_1, \dots, C_k, \mu_1, \dots, \mu_k} \left\{ \sum_{k=1}^K \sum_{i \in C_k} \sum_{j=1}^p (x_{i,j} - \mu_k)^2 \right\}. \quad (4)$$

# Clustering

## K-Means Algorithm



The K-Means algorithm is now able to exploit the new formulation of the optimization problem and iteratively solve for  $\{C_1, \dots, C_k\}$  and  $\{\mu_1, \dots, \mu_k\}$ .

This makes K-Means a greedy algorithm because, in each iteration it chooses optimal values for  $\{C_1, \dots, C_k\}$  and  $\{\mu_1, \dots, \mu_k\}$ .

Convergence of the algorithm is therefore ensured, however we cannot guarantee it will find the global optimum.

# Clustering

## K-Means Algorithm



---

### Algorithm 1: K-Means

---

```
1  Assign each observation to a cluster randomly
2  foreach Cluster do
3      |   Compute the centroid
4  foreach Observation do
5      |   Compute distance to all centroids
6      |   Assign to the closest
7  while Centroids have not changed since last iteration do
8      |   foreach Observation do
9          |   Compute distance to all centroids
10         |   Assign to the closest
11         foreach Cluster do
12             |   Compute the centroid
13 return Clusters
```

---

# Clustering

An example of the K-Means algorithm

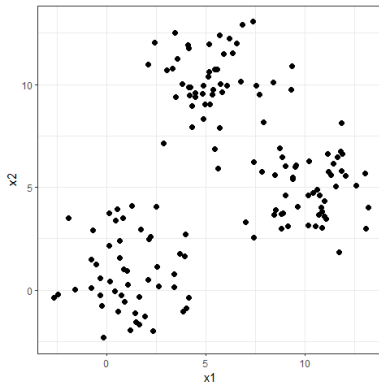


Figure: Iteration 01



# Clustering

An example of the K-Means algorithm

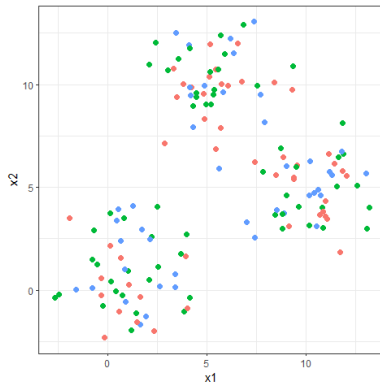


Figure: Iteration 02

# Clustering

An example of the K-Means algorithm

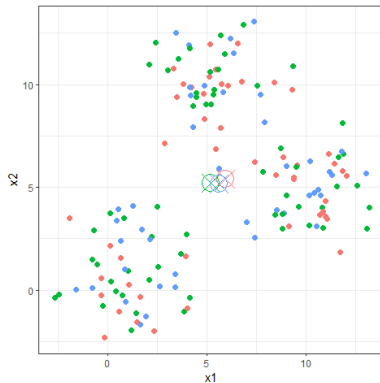


Figure: Iteration 03

# Clustering

An example of the K-Means algorithm

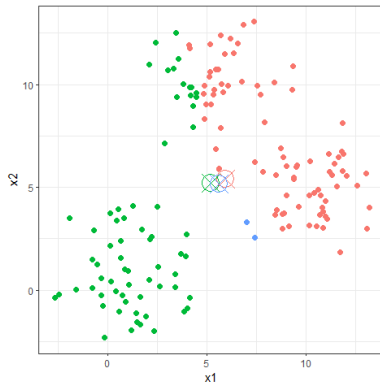


Figure: Iteration 04

# Clustering

An example of the K-Means algorithm

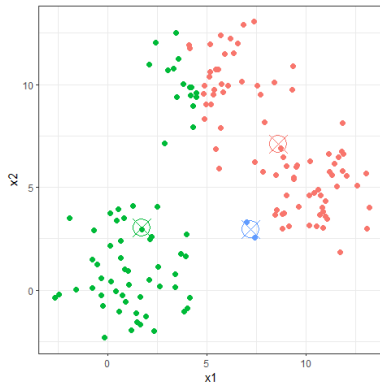


Figure: Iteration 05

# Clustering

An example of the K-Means algorithm

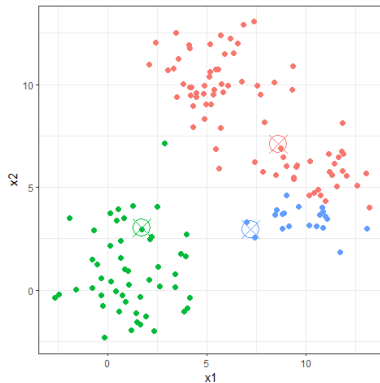


Figure: Iteration 06

# Clustering

An example of the K-Means algorithm

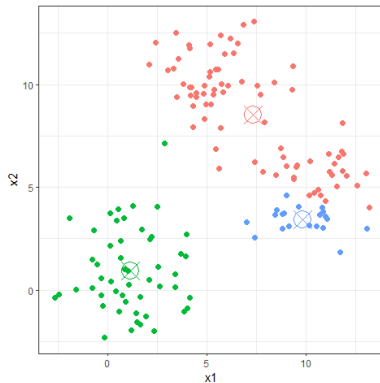


Figure: Iteration 07

# Clustering

An example of the K-Means algorithm

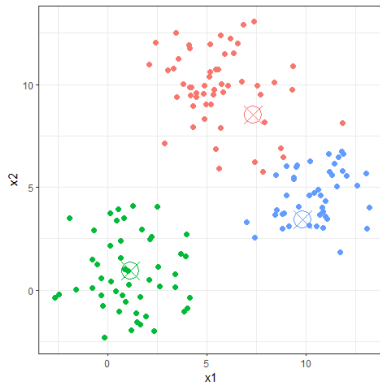


Figure: Iteration 08

# Clustering

An example of the K-Means algorithm

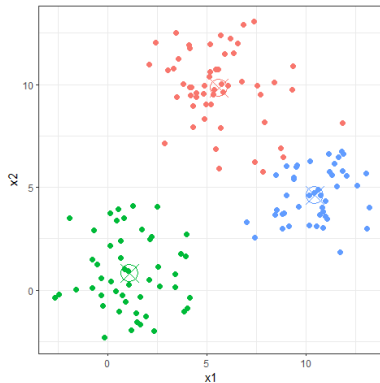


Figure: Iteration 09



# Clustering

An example of the K-Means algorithm

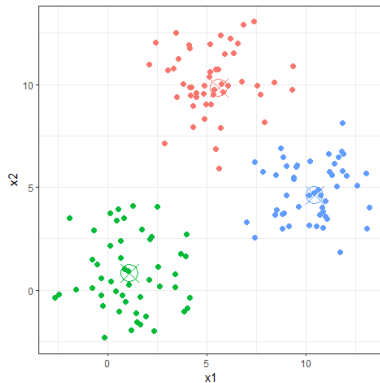


Figure: Iteration 10

# Clustering

An example of the K-Means algorithm



Figure: Iteration 11

# Clustering

An example of the K-Means algorithm



Figure: Iteration 12

# Clustering

An example of the K-Means algorithm

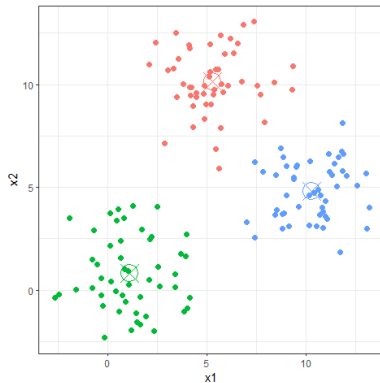


Figure: Iteration 13

# Clustering

## Hierarchical Clustering



### Todo

- ▶ Introduction
- ▶ Type - Agglomerative vs Divise
- ▶ Pseudocode for algorithm, or just in words
- ▶ Visualization with dendogram
- ▶ Linkage types - (complete, single, average, centroid)

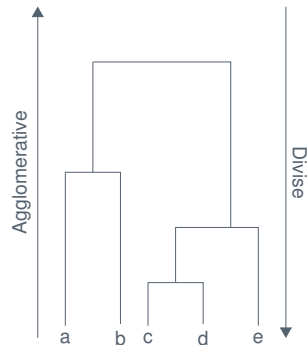
# Clustering

## Hierarchical Clustering

Hierarchical clustering is a method which seeks to build a hierarchy of clusters.

In general there are two approaches to obtain this hierarchy, starting from the bottom or from the top (so to speak):

1. **Agglomerative:** This is the bottom up approach, it starts by having each observation in its own cluster and then it merges the most similar together.
2. **Divise:** This is the top down approach, it start by putting all observations in one cluster and then splits recursively to obtain more and more homogenous clusters.



# Clustering

## Linkage



In order to quantify the similarity between observations we need to establish a dissimilarity measure. However there are many ways to define dissimilarity, or linkage, between groups of observations.

1. **Single:** The smallest pairwise dissimilarity.
2. **Complete:** The largest pairwise dissimilarity.
3. **Average:** The average dissimilarity between observations.
4. **Centroid:** The dissimilarity between centroids of clusters.

# Shrinkage

## Overview



Write theory about ridge, lasso and elastic net - including coordinate descent etc.

Find an example to compare the 3 methods on



# Shrinkage

## Variable Selection and Regularization



Because it is often cheaper to obtain multiple observations from a few samples than to obtain more samples, thus increasing the number of explanatory variables in a regression model, a linear regression model would be prone to increasing variance.

When extending linear regression to multiple explanatory variables the main objectives is:

- ▶ **Model Interpretability:** Models with fewer variables are often more easy to interpret results from and are therefore better to use for decision making.
- ▶ **Prediction Accuracy:** If by introducing some bias we are able to dramatically improve prediction accuracy, this would be worth considering (bias-variance tradeoff).

# Shrinkage

## Variable Selection and Regularization



There are multiple tools we can use in this pursuit

- ▶ **Subset Selection:** Works by fitting lots of models with different combinations of predictors. Then we can find out which variables are most related to the response and we can select these.
- ▶ **Dimensionality Reduction:** Works by projecting explanatory variables into a smaller dimensional space and use these projections as predictors.
- ▶ **Shrinkage Methods:** Works by fitting a model using all predictors while shrinking coefficients towards zero, to reduce variance. Some shrinkage methods can also perform variable selection by shrinking coefficients to exactly zero.

# Shrinkage

## Variable Selection and Regularization



This presentation will focus on shrinkage methods. The two main shrinkage methods are

- ▶ **Ridge Regression**
- ▶ **Lasso**

They both penalize the “size” of the estimated parameters, however they differ in the way they quantify the “size”. Ridge regression penalizes the  $\ell_2$  norm while Lasso penalizes the  $\ell_1$  norm.

Therefore we can also refer to the two methods as  $\ell_1$ - and  $\ell_2$ -regularizations.

# Shrinkage

## Ridge Regression



The expression that ridge regression looks to minimize is the following

$$\underbrace{\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2}_{RSS} + \lambda \underbrace{\sum_{j=1}^p \beta_j^2}_{Penalty}, \quad (5)$$

where  $\lambda$  is a parameter that needs to be determined separately, this is usually done by cross validation.

# Shrinkage

## Lasso



The expression that lasso looks to minimize is the following

$$\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j|,$$

where  $\lambda$ , again, is a parameter that needs to be determined separately.

# Shrinkage

## Elastic Net



The method is a compromise between Ridge Regression and Lasso as it minimizes

$$\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p (\alpha |\beta_j| + (1 - \alpha) \beta_j^2),$$

note that here we have the additional parameter  $\alpha$  that determines the weighting of the two norms.

# Classification

## Overview



Add something about logit and naive bayes, very brief

Maybe include example with an ROC curve

# Classification

## Introduction

If the response variable is categorical (qualitative), i.e. it is of the form  $y \in \{1, \dots, L\}$ . Then a linear model of the form

$$y = \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p,$$

is generally not a good approach to take as it could predict invalid values and for certain types of categorical data there might not be a clear ordering.

Therefore when dealing with categorical variable the aim of the model is to predict the probability that an observation belongs to a certain category, rather than the category itself.



# Classification

## Linear Discriminant Analysis (LDA)

Suppose we have a dataset with a response variable  $y \in \{0, \dots, L\}$ , explanatory variables  $x_1, \dots, x_p$  and that we would like to model

$$P(y = k|X) = \underbrace{\frac{P(y = k)P(X|y = k)}{P(X)}}_{\text{Bayes Theorem}} = \frac{\pi_k f_k(x)}{\sum_{i=1}^K f_i(x)}, \quad (6)$$

where  $\pi_k = P(y = k)$  and  $f_k(x) = P(X|y = k)$ .

If we use the proportion of observations in the dataset that belong to class  $k$  as an estimate for  $\pi_k$ , then we just need to model  $f_k(x) = P(X|y = k)$ .

In other words we need to make some assumption on the distribution of  $f_k(x)$ .

# Classification

## Linear Discriminant Analysis (LDA)

The assumption made in LDA is that each  $f_k(x)$  come from a multivariate normal distribution, i.e.

$$f_k(x) = \frac{1}{(2\pi)^{p/2} |\Sigma_k|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu_k)^\top \Sigma_k^{-1} (x - \mu_k) \right\}. \quad (7)$$

Another assumption made in LDA is that  $\Sigma_k = \Sigma \forall k \in \{1, \dots, K\}$ , in other words all classes will have the same variance-covariance matrix.

# Classification

## Linear Discriminant Analysis (LDA)

When we classify a new observation,  $X_0$ , we simply find the category with the highest probability,  $P(y = k|X_0)$  for  $k = 1, \dots, K$ .

In other words we want to find the  $k$  such that  $P(y = k|X)$  is maximized. Since the logarithm is an increasing function we know that the  $k$  which maximizes  $P(y = k|X)$  also maximizes the following

$$\log(P(y = k|X)) = \log(\pi_k) + \log(f_k(x)) - \underbrace{\log\left(\sum_{i=1}^K \pi_i f_i(x)\right)}_{\text{Does not depend on } k}, \quad (8)$$

The last term is identical across categories and we drop this term for the maximization problem.

# Classification

## Linear Discriminant Analysis (LDA)

So now we have the following

$$\log(\pi_k) + \log(f_k(x)), \quad (9)$$

but let us take a look at the second term. By the normality assumption we have that

$$f_k(x) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu)^\top \Sigma^{-1} (x - \mu_k) \right\}. \quad (10)$$

Thus we can write (9) as

$$\log(\pi_k) + \underbrace{\log((2\pi)^{p/2} |\Sigma|^{1/2})}_{\text{Does not depend on } k} - \frac{1}{2} (x - \mu)^\top \Sigma^{-1} (x - \mu_k), \quad (11)$$

and subsequently drop another term that is identical across categories.

# Classification

## Linear Discriminant Analysis (LDA)

So now we have the following

$$\log(\pi_k) - \frac{1}{2}(x - \mu)^{\top} \Sigma^{-1}(x - \mu_k), \quad (12)$$

again taking a look at the second term

$$(x - \mu_k)^{\top} \Sigma^{-1}(x - \mu_k) = \underbrace{x^{\top} \Sigma^{-1} x}_{\text{Does not depend on } k} - x^{\top} \Sigma^{-1} \mu_k - \mu_k^{\top} \Sigma^{-1} x + \mu_k^{\top} \Sigma^{-1} \mu_k, \quad (13)$$

and furthermore we have that  $x^{\top} \Sigma^{-1} \mu_k = \mu_k^{\top} \Sigma^{-1} x$  because both are scalars and one is just the transpose of the other.

# Classification

## Linear Discriminant Analysis (LDA)

So we end up with the following expression, which is called the **linear discriminant function**

$$\delta_k(x) = \log(\pi_k) + x^\top \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^\top \Sigma^{-1} \mu_k, \quad (14)$$

this is what we will use to classify observations. Note the expression is linear in  $x$  and therefore we call it linear discriminant analysis.

# Classification

## Linear Discriminant Analysis (LDA)

In practice the parameters are estimated by

$$\hat{\pi}_k = \frac{n_k}{n}, \quad \hat{\mu}_k = \frac{1}{n_k} \sum_{y_i=k} x_i, \quad \hat{\Sigma} = \frac{1}{n-K} \sum_{k=1}^K \sum_{y_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^{\top}, \quad (15)$$

i.e. the proportion of observations in class  $k$ , the mean for class  $k$  and the mean of the variances respectively.

# Classification

## Quadratic Discriminant Analysis (QDA)

Imagine now that we relax the LDA assumption that  $\Sigma_k = \Sigma \forall k \in \{1, \dots, K\}$  and allow classes to have their own variance-covariance matrix.

This implies that some of the simplification we performed before no longer holds and the discriminant function will now take the form

$$\delta_k(x) = -\frac{1}{2}x^\top \Sigma_k x + \log(\pi_k) + x^\top \Sigma_k^{-1} \mu_k - \frac{1}{2} \mu_k^\top \Sigma_k^{-1} \mu_k - \frac{1}{2} \log |\Sigma_k|, \quad (16)$$

note that first term is quadratic in  $x$ .

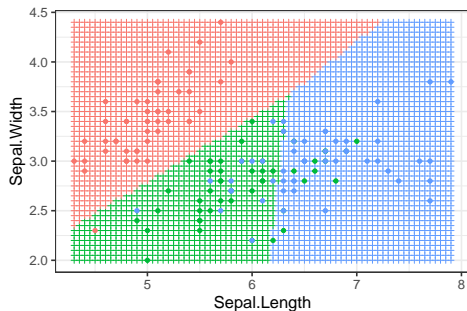
Another difference in the estimation of the covariance matrices, because we relaxed the assumption we must estimate a variance-covariance matrix for each class

$$\hat{\Sigma}_k = \frac{1}{n_k - 1} \sum_{y_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^\top. \quad (17)$$

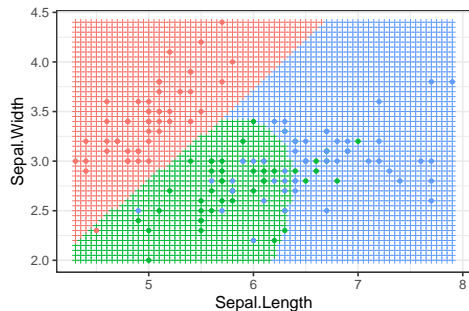


# Classification

Example using the Iris dataset



(a) LDA classification regions.



(b) QDA classification regions.

**Figure:** Classification regions for LDA and QDA performed on the *Iris* dataset, considering only the variables *sepal width* and *sepal length*. Note that the LDA boundaries are linear while QDA include a non-linear boundary.

# Classification

## Naive Bayes



# Trees

## Overview



# Trees

## Introduction



Tree-based methods are supervised learning methods which can be used for both regression and classification.

They work by segmenting the space into a number of simple regions and then, typically, use the mean of the region to make predictions.

Because the segmentation of the space can be summarized in a tree, we call these methods tree-based.



# Trees

## Classification and Regression Trees (CART)

The purpose of the CART algorithm is to decide on the split points. This is done in a greedy way one split at a time, as the task of checking all possible combinations of splits would be computationally infeasible.

# Trees

## Classification and Regression Trees (CART)

Consider the following model (non-binary)

$$y = 1 + 2x_1 + 3x_2 + \varepsilon, \quad (18)$$

where  $x_1, x_2 \sim N(0, 1)$  and  $\varepsilon \sim N(0, 1/2)$ . We can then introduce the variables  $\tilde{x}_1, \tilde{x}_2$  given by

$$\tilde{x}_1 = \begin{cases} 1 & \text{for } x_1 > 0 \\ 0 & \text{for } x_1 \leq 0 \end{cases}, \quad \tilde{x}_2 = \begin{cases} 1 & \text{for } x_2 > 0 \\ 0 & \text{for } x_2 \leq 0 \end{cases}, \quad (19)$$

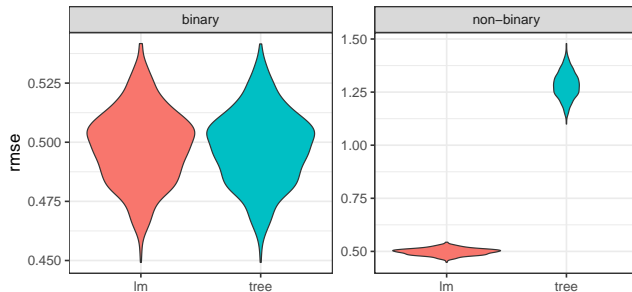
and propose a separate model (binary)

$$\tilde{y} = 1 + 2\tilde{x}_1 + 3\tilde{x}_2 + \varepsilon. \quad (20)$$

We can then compare how the CART algorithm compares to linear regression on both the binary and non-binary model.

# Trees

## Classification and Regression Trees (CART)



**Figure:** Comparing linear regression (`lm` from base R) and CART (`tree` from the `tree` package) on 1000 repetitions of 500 observations. We see that when the data is generated from a step function it's a tie between `tree` and `lm`, but when the data is linear the CART understandably struggles compared to linear regression.

# Trees

## Bagging





# Trees

## Random Forest



# Trees

## Boosting



# Support Vector Machines

## Overview



# Neural Networks

## Overview



# Neural Networks

## Perceptron



**Backpropagation:** For a given loss function  $L$  we look for  $\frac{\partial L}{\partial w_i}$ . We start with initial values for the weights, which we shall denote  $w_{old}$ . Then we update the weights by  $w_{new} = w_{old} - \eta \frac{\partial L}{\partial w}$ . One iteration is called an **epoch**.

